Introduction

Le mot logique provient du grec logos qui signifie "science de la raison". Il est présenté dans tous les domaines de l'informatique :

- Architecture des ordinateurs.
- Programmation.
- Traitement automatique des langues.
- Intelligence artificielle.

On s'intéressera principalement à la logique classique, qui est la logique utilisée pour les mathématiques, et forme la base de presque toutes les autres logiques.

Nous allons nous intéresser aux fragments suivants :

- la logique propositionnelle ;
- la logique du premier ordre.

1. Calcul propositionnel

1.1. Syntaxe du calcul propositionnel

En mathématique, une expression bien formée ou proposition est une expression qui a du sens et qui peut être vraie ou fausse.

La logique est basée sur le principe de non-contradiction. Ce principe dit qu'une expression bien formée ne peut pas être vraie et fausse à la fois.

Le langage du calcul propositionnel est formé de :

- symboles propositionnels Prop = {p, q,r...} ;
- connecteurs logiques {┐, ∧, ∨, →, ↔} ;
- symboles auxiliaires : parenthèses ().

Remarque : Dans la littérature logique on utilise plusieurs synonymes pour symbole propositionnel ; ainsi variable propositionnelle, proposition atomique, formule atomique, ou encore atome sont tous des synonymes de symbole propositionnel.

L'ensemble Fcp des formules du calcul propositionnel est le plus petit ensemble tel que :

- tout symbole propositionnel est une formule ;
- si p est une formule alors $\neg p$ est une formule ;
- si p, q sont des formules alors p^q ,p∨q ,p→q et p↔q sont des formules.

Les symboles auxiliaires ne sont utilisés que pour lever les ambiguïtés possibles : par exemple, la formule p∨q^r est ambiguë, car elle peut se lire de deux façons différentes, ((p∨q)^r) ou bien (p∨(q ∧ r)).

1.2. Sémantique du calcul propositionnel

Il faut maintenant un moyen de déterminer si une formule est vraie ou fausse. La première étape est de donner une valeur de vérité aux propositions atomiques. L'évaluation d'une formule, dépend donc des valeurs choisies pour les symboles propositionnels. Ces valeurs sont données par une valuation.

1.2.1.Définition (Valuation). Une valuation est une application de Prop dans {0, 1}.La valeur 0 désigne le "faux" et la valeur 1 désigne le "vrai".

1.2.2.Définition (Valeur d'une formule).

— $v(\neg\varphi) = 1$ ssi $v(\varphi) = 0$;
— $v(\varphi \vee \psi) = 1$ ssi $v(\varphi) = 1$ ou $v(\psi) = 1$;
— $v(\varphi \wedge \psi) = 1$ ssi $v(\varphi) = 1$ et $v(\psi) = 1$;
— $v(\varphi \Rightarrow \psi) = 0$ ssi $v(\varphi) = 1$ et $v(\psi) = 0$.

1.2.3.Définition (Valeur d'une formule).

— $v(\neg\varphi) = 1 - v(\varphi)$;
— $v(\varphi \vee \psi) = \max(v(\varphi), v(\psi))$;
— $v(\varphi \wedge \psi) = \min(v(\varphi), v(\psi))$;
— $v(\varphi \Rightarrow \psi) = v(\neg\varphi \vee \psi)$.

Ce qui correspond aux tables de vérité des connecteurs logiques.

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

p	$\neg p$
0	1
1	0

p	q	$p \Rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

p	q	$p \Leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

1.2.4.Définition (Modèle d'une formule)

Un modèle de p est une valuation v telle que v(p)=1.On note mod(p') l'ensemble des modèles de p.

Définition (Satisfaisabilité)

Une formule F est **satisfaisable (**ou **consistante,** ou **cohérente)** si elle admet un modèle (s'il existe une valuation v telle que v(F)=1).

Définition (InSatisfaisabilité)

Une formule F est **insatisfaisable (**ou **inconsistante,** ou **incohérente)** si elle n'admet aucun modèle.

Définition(Tautologie)

Une formule F est une **tautologie (ou valide)** si v(F)=1 pour toute valuation v.On note $\models$F pour dire que F est une tautologie.

Un exemple de tautologie est $F \vee \neg F$.

Définition 3.4 (Equivalence)

Deux formules F et G sont dites équivalentes si pour toute valuation v, v(F) = v(G). On écrit dans ce cas F≡G

Exemple : Les formules p et $\neg\neg p$ sont équivalentes.

Remarque Il est important de bien comprendre que ≡ est un symbole que l'on utilise pour écrire une relation entre deux formules, mais que F≡G n'est pas une formule propositionnelle.

Exercice : Montrer que cette formule est une tautologie ?

$$F \rightarrow F$$

Exercice : Montrer que pour toute formule F on a les équivalences suivantes ?

$$(F \vee F) \equiv F$$

$$(F \wedge F) \equiv F$$

Propriété (Formules équivalentes) : Soit A, B et C trois formules bien formées.

Implication

- $A \rightarrow B \equiv \neg A \vee B$
- $\neg (A \rightarrow B) \equiv A \wedge \neg B$
- $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

Idempotence

- $A \wedge A \equiv A$
- $A \vee A \equiv A$

Commutativité

- $A \vee B \equiv B \vee A$
- $A \wedge B \equiv B \wedge A$

Associativité

- $(A \vee B) \vee C \equiv A \vee (B \vee C)$
- $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$

Distributivité

- $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
- $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$

Élément neutre

- $A \vee F \equiv A$
- $A \wedge T \equiv A$
- $A \vee T \equiv T$
- $A \wedge F \equiv F$

Complémentarité

- $A \vee \neg A \equiv T$
- $A \wedge \neg A \equiv F$

Involution

- $\neg\neg A \equiv A$

De Morgan

- $\neg (A \vee B) \equiv \neg A \wedge \neg B$
- $\neg (A \wedge B) \equiv \neg A \vee \neg B$

Absorption

- $A \vee (\neg A \wedge B) \equiv A \vee B$
- $A \wedge (\neg A \vee B) \equiv A \wedge B$
- $A \vee (A \wedge B) \equiv A$
- $A \wedge (A \vee B) \equiv A$

3.5. Formes normales

3.5.1 Formes normales conjonctives et disjonctives

On cherche souvent à ramener les formules sous une forme équivalente la plus simple possible.

Définition (Littéral, monôme, clause)

- Un littéral est une variable ou la négation d'une variable.
- Un monôme est une conjonction de littéraux.
- clause est une disjonction de littéraux.

Définition : Une forme normale disjonctive est une disjonction $F1 \vee F2 \ldots \vee F_k$ de k formules, $k \geq 1$ où chaque formule F_i, $1 \leq i \leq k$ est une conjonction $G1 \wedge G2 \ldots . \wedge G_\lambda$ de λ littéraux.

Exemple : Les formules suivantes sont des formes normales disjonctives

$$((p \wedge q) \vee (\neg p \wedge \neg q)$$

$$((p \wedge q \wedge \neg r) \vee (q \wedge \neg p)$$

$$(p \wedge \neg q)$$

Définition : Une forme normale conjonctive est une conjonction $F1 \wedge F2 \ldots \wedge F_k$ de k formules, $k \geq 1$ où chaque formule F_i, $1 \leq i \leq k$ est une disjonction $G1 \vee G2 \ldots . \vee G_\lambda$ de λ littéraux.

Exemple : Les formules suivantes sont des formes normales conjonctives

$$(\neg p \vee q) \wedge (p \vee \neg q)$$

$$(\neg p \vee q) \wedge \neg r$$

Théorème : Toute formule sur un nombre fini de variables propositionnelles est équivalente à une formule en forme normale conjonctive.

Théorème : Toute formule sur un nombre fini de variables propositionnelles est équivalente à une formule en forme normale disjonctive.

3.5.2 Méthodes de transformation

La méthode consiste à transformer la formule par équivalences successives à l'aide des règles suivantes appliquées dans cet ordre :

1. élimination des connecteurs $\leftrightarrow$ par :

$$F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$$

2. élimination des connecteurs $\rightarrow$ par :

$$(F \rightarrow G) \equiv (\neg F \vee G)$$

3. entrée des négations le plus à l'intérieur possible :

$$\neg(F \wedge G) \equiv (\neg F \vee \neg G)$$

$$\neg(F \vee G) \equiv (\neg F \wedge \neg G)$$

4. distributivité de $\wedge$ et $\vee$ l'un par rapport à l'autre :

$$F \wedge (G \vee H) \equiv ((F \wedge G) \vee (F \wedge H)$$

$$F \vee (G \wedge H) \equiv ((F \vee G) \wedge (F \vee H)$$

Exemple : Mettre la formule $\neg(p \Rightarrow (q \Rightarrow r)) \vee (r \Rightarrow q)$ sous forme normal disjonctive et conjonctive.

On utilise les équivalences successives

$$\neg(\neg p \vee (\neg q \vee r)) \vee (\neg r \vee q)$$
$$(p \wedge \neg(\neg q \vee r)) \vee (\neg r \vee q)$$
$$(p \wedge q \wedge \neg r) \vee (\neg r \vee q)$$

Qui est une forme normale disjonctive.

$$(p \wedge q \wedge \neg r) \vee (\neg r \vee q)$$
$$(p \wedge \neg r \vee q) \wedge (\neg r \vee q)$$

3.5.3 Substitutions

Il faut définir ce que signifie remplacer p par G dans une formule F, noté F (G/p).

La substitution de p par G dans F est définie par induction sur la formule F :

- Si F est la variable propositionnelle p, alors F(G/p) est la formule G.
- Si F est la variable propositionnelle q , avec q $\neq$ p ,alors F(G/p) est la formule F.
- Si F est de la forme $\neg H$,alors F (G/p) est de la formule $\neg H(G/p)$
- Si F est de la forme ($F_1 \vee F_2$), alors F (G/p) est la formule (F_1(G/p))$\vee$ (F_2(G/p)).
- Si F est de la forme ($F_1 \wedge F_2$), alors F (G/p) est la formule (F_1(G/p))$\wedge$ (F_2(G/p)).
- Si F est de la forme ($F_1 \Rightarrow F_2$), alors F (G/p) est la formule (F_1(G/p)) $\Rightarrow$ (F_2(G/p)).
- Si F est de la forme ($F_1 \Leftrightarrow F_2$), alors F (G/p) est la formule (F_1(G/p)) $\Leftrightarrow$ (F_2(G/p)).

Définition (Instance) :On dit qu'une formule F est une instance d'une formule G si F s'obtient en substituant certaines variables propositionnelles de G par des formules Fi.

Exemple : La formule $((C \Rightarrow D) \Rightarrow (\neg A \Rightarrow (C \Rightarrow D)))$ est une instance de $(A \Rightarrow (B \Rightarrow A))$ en prenant $(C \Rightarrow D)$ pour A, et $\neg A$ pour B.

Définition d'un système formel

Un système formel définit des règles de calcul entre formules qui simulent le raisonnement, ce qui permet de déduire de nouvelles formules. Il est défini par :

- ✓ Le langage et les formules sur ce langage.
- ✓ Les axiomes : formules ou schémas de formules supposées vraies.
- ✓ Les règles d'inférence : règles permettant de déduire une nouvelle formule.

5. Le théorème de complétude pour la logique booléenne

On introduit la notion de preuve par coupure en logique booléenne, et on établit le théorème de complétude affirmant que toute formule valide est prouvable à partir d'une famille finie explicite d'axiome simple.

4.1. Preuves par coupure « Démonstration à la Frege et Hilbert »

On introduit la notion de preuve par coupure (ou modus ponens).

Preuves formelles, premières propriétés

LES AXIOMES

On prend trois schémas d'axiomes:

(A1) : Pour toutes formules propositionnelles A, B

$(A \to (B \to A))$

(A2) : pour toutes formules propositionnelles A, B, C

$((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$

(A3) : pour toutes formules propositionnelles A, B

$(\neg B \to \neg A) \to (A \to B)$

Une RÈGLE de déduction:

- le Modus Ponens: pour toutes formules propositionnelles A, B, de $\{A, (A \to B)\}$ on déduit B.

Conséquence Logique

Définition: Soit F un ensemble de formules propositionnelles et A une formule propositionnelle. On dit que A est conséquence logique, ou conséquence formelle de F, noté $F \models A$, s'il existe une démonstration formelle (ou preuve formelle) de A à partir de F, c'est-à-dire, s'il existe une suite finie $\{A_1 , . . . ,A_k \}$ de formules propositionnelles,

telles que :

- $A_k = A$
- pour chaque $i \leq k$, l'une des trois conditions suivantes est remplie
- soit A_i est une formule de F,
- soit A_i est un axiome.
- soit A_i a été déduit par Modus Ponens de deux formules A_l et A_m

<u>Exemples de preuves formelles</u>

Preuve de $\vdash A \to A$

f_0 $A \to (A\to A)$…………….. A_1

f_1 $(A \to (A \to A) \to A)\to (A\to (A \to A) \to (A\to A))$ ………………….A_2

f_2 $A \to (A \to A) \to A$……………..A_1

f_3 $(A\to (A \to A) \to (A\to A)$…………… Modus Ponens($f_1$; f_2)

$(A \rightarrow A)$ …………..Modus Ponens (f_0 ;f_3).

Lemmes : F est un ensemble de formules, A et B sont des formules:

Lemme 2.2 Si F $\vdash$A et F $\vdash(A \rightarrow B)$, alors F $\vdash$B.

Lemme 2.5 Si F $\vdash(A \rightarrow B)$, alors F ∪ {A} $\vdash$B.

Lemme 2.6 (Lemme de déduction)

Si F ∪ {A} $\vdash$B , alors F $\vdash(A \rightarrow B)$.

Lemme 2.8

Pour tout ensemble de formules F, pour toutes formules A et B,

Si F ∪ {A} $\vdash$B et F ∪ {¬A} $\vdash$B, alors F $\vdash$B.

Définition : On dit que Σ est **contradictoire** s'il existe une formule A telle que F $\vdash$A et F $\vdash$¬A.

4.2. La méthode de résolution

La règle de résolution

Théorème : Le schéma suivant est bien une règle d'inférence:

$\{X \vee A, \neg X \vee B\} \models A \vee B$ (règle de résolution)

Démonstration:

De manière informelle, on peut dire la chose suivante: si on a X, alors on n'a pas ¬X, donc ayant ¬X ∨ B, on a nécessairement B, donc A ∨ B. De même si on a ¬X, on n'a pas X, donc ayant X ∨ A, on a nécessairement A donc A ∨ B. Dans les deux cas on a ainsi A ∨ B, comme on est forcément dans un de ces deux cas..., on a bien nécessairement A ∨ B.

Ou bien aussi: noter que $(\neg X \vee B) \equiv (X \Rightarrow B)$ et que $(A \vee X) \equiv (\neg A \Rightarrow X)$, on a ainsi: $\{\neg A \Rightarrow X, X \Rightarrow B\}$, qui donne par règle du syllogisme: $\neg A \Rightarrow B$, équivalent à A ∨ B.

Proposition : la règle $\{X, \neg X\} \models$ □ est un cas particulier de la règle de résolution. (avec A et B remplacés par□ .)

Application de la règle de résolution à un ensemble de clauses

• **clause:** une disjonction de littéraux (p∨q∨¬r, ¬p∨¬q∨¬r, p∨q, p, ¬p etc.)

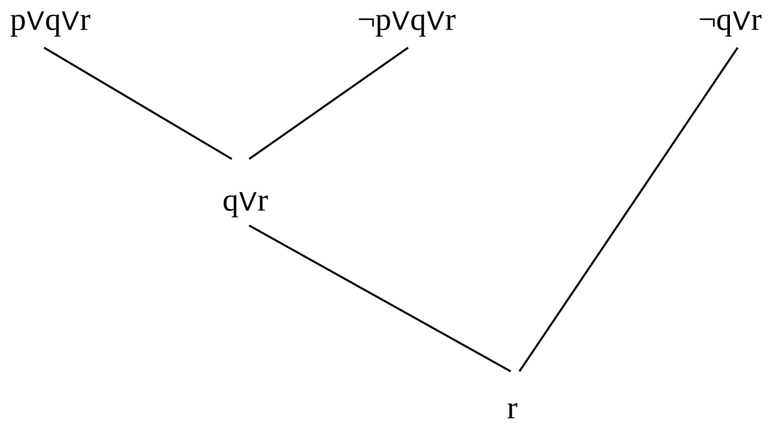
Noter qu'une disjonction vide est, comme dans tous les cas similaires, l'élément neutre de la disjonction, à savoir: □ . C'est la raison pour laquelle on appellera souvent □ : **la clause vide.**

• **résolvante de deux clauses:** soit deux clauses, s1 et s2, telles que le littéral l figure dans l'une et le littéral ¬l figure dans l'autre (on écrira par exemple: $l \in s_1$, $\neg l \in s_2$), on appelle résolvante de s1 et s2 la clause obtenue en supprimant le littéral l de l'une et le littéral ¬l de l'autre et en réunissant tout ce qui reste en une seule clause.

Exemple: si $s_1 = \neg p \lor q \lor r$ et $s_2 = p \lor q \lor s$, alors $Res(s_1, s_2) = q \lor r \lor s$ (on supprimera aussi en même temps les littéraux redondants).

Exemple:

$\{p \lor q \lor r, \neg p \lor q \lor r, \neg q \lor r\} \models r$



4-3.Définition: Résolution + réfutation : partir d'un ensemble de clauses contenant la négation de la proposition à démontrer et appliquer la règle de résolution un certain nombre de fois, en enrichissant chaque fois l'ensemble de départ avec les déductions qui sont faites, jusqu'à obtenir □ si possible.

Exemple:

Ensemble de départ: {pVqVr, ¬pVqVr, **¬qVr, ¬r**}

1er pas: {**pVqVr**, **¬pVqVr**, ¬qVr, ¬r, ¬q}

2eme pas : {pVqVr, ¬pVqVr, **¬qVr**, ¬r, ¬q, **qVr**}

3eme pas: {pVqVr, ¬pVqVr, ¬qVr, **¬r**, ¬q, qVr, **r**}

4eme pas: {pVqVr, ¬pVqVr, ¬qVr, ¬r, ¬q, qV r, r, □ }

Stop! On a trouvé□ .

4.4. Clauses de Horn

On peut avoir un algorithme ayant de meilleures qualités en se restreignant à une famille de clauses particulières appelées clauses de Horn.

Définition : un littéral est dit positif s'il consiste en une lettre (sans négation), il est dit négatif dans le cas contraire.

Définition : on appelle clause de Horn toute clause qui possède au plus un littéral positif.

Exemples de clauses de Horn:

$\neg p \lor \neg q \lor r$

r

$\neg p \lor \neg q \lor \neg r$

Définition : on appellera règle, ou clause de Horn stricte, toute clause qui possède exactement un littéral positif, on appellera clause de Horn négative toute clause sans littéral positif, et on appellera fait, ou clause unitaire positive toute clause consistant en un littéral positif.

Algorithme de démonstration d'inconsistance d'un ensemble de clauses de Horn:

Soit S l'ensemble de clauses

Tant que $\square \notin S$

- choisir p et c tels que:
  - p soit une clause unitaire positive de S.
  - c soit une clause de S contenant ¬p;
- calculer la résolvante r de p et c;
- remplacer S par (S–{c})∪{r}

S'il n'y a plus de nouveau choix possible et que $\square \notin S$, alors S est consistant,

Si $\square \in S$, alors S est inconsistant.

Exemple :

S= {p ∨ ¬r ∨ ¬t, q, r, t ∨ ¬p ∨ ¬r, t ∨ ¬q, ¬p ∨ ¬q ∨ ¬r}

1er pas : {**p ∨ ¬r ∨ ¬t**, q, **r**, t ∨ ¬p ∨ ¬r, t ∨ ¬q, ¬p ∨ ¬q ∨ ¬r} donne:

{p ∨ ¬t, q, r, t ∨ ¬p ∨ ¬r, t ∨ ¬q, ¬p ∨ ¬q ∨ ¬r}

2eme pas : {p ∨ ¬t, q, **r**, **t ∨ ¬p ∨ ¬r**, t ∨ ¬q, ¬p ∨ ¬q ∨ ¬r} donne:

{p ∨ ¬t, q, r, t ∨ ¬p, t ∨ ¬q, ¬p ∨ ¬q ∨ ¬r}

3eme pas : {p ∨ ¬t, **q**, r, t ∨ ¬p, **t ∨ ¬q**, ¬p ∨ ¬q ∨ ¬r} donne:

{p ∨ ¬t, q, r, t ∨ ¬p, t , ¬p ∨ ¬q ∨ ¬r}

4eme pas : {**p ∨ ¬t**, q, r, t ∨ ¬p, **t** , ¬p ∨ ¬q ∨ ¬r} donne:

{p, q, r, t ∨ ¬p, t , ¬p ∨ ¬q ∨ ¬r}

5eme pas : {**p**, q, r, **t ∨ ¬p**, t , ¬p ∨ ¬q ∨ ¬r} donne:

{p, q, r, t , t , ¬p ∨ ¬q ∨ ¬r}

par élimination des littéraux redondants:

{p, q, r, t , ¬p ∨ ¬q ∨ ¬r}, qui donne:

6eme pas : {p, q, r, t , ¬q ∨ ¬r}

7eme pas : {p, **q**, r, t , **¬q ∨ ¬r**} donne:

{p, q, r, t , ¬r}

8eme pas : {p, q, **r**, t , **¬r**} donne:

{p, q, r, t , □ }

stop !

On a bien prouvé que l'ensemble était inconsistant.

Exercice :

1. mettre ces formules sous la forme des clauses :
   a. (q→(p∧r))∨p
   b. ((p∨p)→r)∧(p→t)
   c. (p↔p)→((p→q)∧(q→p))

Solution :

a. (q→(p∧r))∨p

$$(\neg q \vee (p \wedge r)) \vee p$$

$$(\neg q \vee p) \wedge (\neg q \vee r) \vee p$$

$$(\neg q \vee p) \wedge (\neg q \vee r \vee p)$$

b. ((p∨p)→r)∧(p→t)

$(\neg(p \vee p) \vee r) \wedge (\neg p \vee t)$

$((\neg p \wedge \neg p) \vee r) \wedge (\neg p \vee t)$

$(\neg p \vee r) \wedge (\neg p \vee r) \wedge (\neg p \vee t)$

$(\neg p \vee r) \wedge (\neg p \vee r)$

d. (p↔p)→((p→q)∧(q→p))

$\neg((p \Rightarrow p) \wedge (p \Rightarrow p)) \vee ((p \Rightarrow q) \wedge (q \Rightarrow p))$

$\neg((\neg p \vee p) \wedge (\neg p \vee p)) \vee ((\neg p \vee q) \wedge (\neg q \vee p))$

$(\neg(\neg p \vee p) \vee \neg(\neg p \vee p)) \vee ((\neg p \vee q) \wedge (\neg q \vee p))$

$((p \wedge \neg p) \vee (p \wedge \neg p)) \vee ((\neg p \vee q) \wedge (\neg q \vee p))$

$((p \wedge \neg p) \vee (p \wedge \neg p) \vee (\neg p \vee q)) \wedge ((p \wedge \neg p) \vee (p \wedge \neg p) \vee (\neg q \vee p))$

$((p \wedge \neg p) \vee (\neg p \vee q)) \wedge ((p \wedge \neg p) \vee (\neg q \vee p))$

$(p \vee \neg p \vee q) \wedge (\neg p \vee \neg p \vee q) \wedge (p \vee \neg q \vee p) \wedge (\neg p \vee \neg q \vee p)$

Complexité de la recherche de modèles

Etant donné une formule avec n variables, il y a 2^n interprétations possibles. Par exemple, une formule à 100 variables possède 1267650600228229401496703205376 interprétations.

Si on essaye toutes les interprétations possibles, la complexité en temps sera donc exponentielle et les temps effectifs de calculs seront énormes (infaisables). Si on teste 1milliard d'interprétations par seconde on en aura pour 40 196 936 841 331 années avec 100 variables.

Or on ne connait pas, à l'heure actuelle, d'algorithme qui fournisse à coup sûr une réponse en temps non exponentiel. Et ce problème est aussi difficile que de nombreux autres pour lesquelles on ne connait pas d'algorithme non exponentiel (on dit que ce problème est NP-complet).

Cependant, il existe des algorithmes, basés sur celui de Davis-Putnam-Logemann-Loveland, qui peuvent trouver une solution rapidement dans de nombreux cas, mais pas toujours .

Utilisation de la FNC pour la satisfiabilité

Algorithme de Davis-Putnam-Logemann-Loveland
On considère une formule ψ qui a été mise en FNC.
L'algorithme consiste essentiellement à
1. repérer les clauses littérales (une seule variable), fixer la valeur de la variable correspondante, mettre à jour les autres clauses.
2. choisir un littéral k (au hasard ou à l'aide d'une heuristique)
3. appliquer l'algorithme sur ψ ∧ k et si on n'a pas trouvé de modèle, l'appliquer sur ψ ∧ ¬(k)
Si à un moment donné la formule n'est composée que de littéraux et qu'elle est consistante, on a trouvé un modèle. Si par contre on a généré une clause vide on est sûr de ne pas trouver de modèle.

Algorithme DPLL

Entrée : une formule F en FNC, un modèle partiel M
Sortie : faux si on n'a pas trouvé de modèle, vrai si on a trouvé + le modèle
Si F est vide :

Retourner vrai et M

Si F contient une clause vide :

Retourner faux

pour toute clause unitaire (x) ou (¬x) de F :

fixer M(x)= v(resp. f)

F = PropagationValeur(x, F)

K := choisir un littéral de F

retourner DPLL(F ∧ K) ou sinon DPLL(F ∧ ¬K)

Propagation des valeurs

Si on a fixé M(x)= v

Supprimer toutes les clauses ou x apparaît positivement

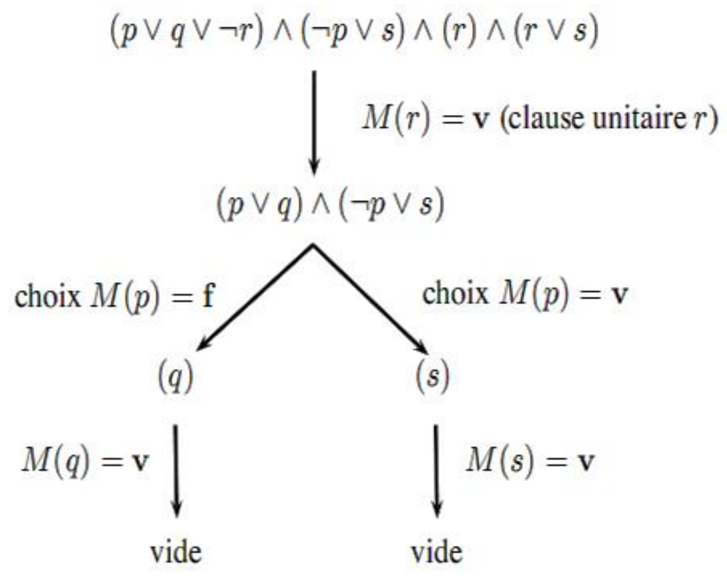
Enlever ¬x de toutes les clauses où il apparaît

Si on a fixé M(x)= f

Supprimer toutes les clauses ou ¬x apparaît

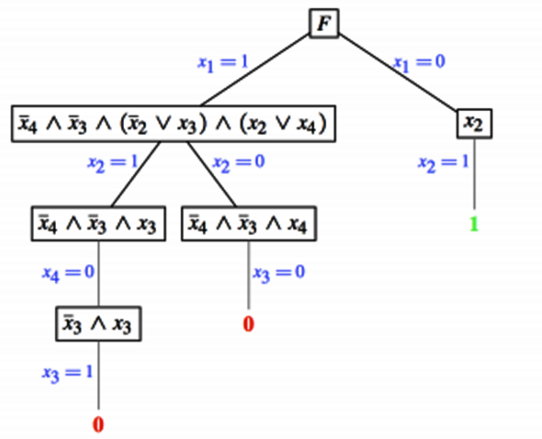
Enlever x de toutes les clauses où il apparaît positivement

Exemple 1



Exemple 2 [1]

$$F = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$$



Déroulement de l'algorithme avec essais-erreurs ($\overline{x}$ signifie $\neg x$)

Algèbre de Boole

Opération. Une autre notation (algébrique)

Formule logique	Algèbre de Boole
$A \vee B$	$A + B$
$A \wedge B$	$A \cdot B$
$\neg A$	$\overline{A}$
f	0
v	1

Donc

p ⇒ q devient $\overline{p} + q$

Quelques formules booléennes

$$0+0=0, 0+1=1+0=1, 1+1=1$$

$$0\cdot 0=0, 0\cdot 1=1\cdot 0=0, 1\cdot 1=1$$

Exercice : Prouvez que f10, f11 et f12 sont des tautologies.

f10 = (p → p)

f11 = ((p → ¬ p) → ¬ p)

f12 = ((p → q) → (¬ q→¬ p))

Pour prouver que f est une tautologie, il faut établir que I(f)est égale à vrai pour toute interprétation. Chaque ligne des preuves devra être justifiée. Utilisez uniquement les rappels sur l'algèbre booléenne et les faits suivants où fa et fb sont des formules logiques :

$I((fa \Rightarrow fb)) = \overline{I(fa)} + I(fb)$,

$I(\neg fa) = \overline{I(fa)}$

Exemple 4.13. On veut prouver

$$\{(p \vee t) \Rightarrow q, r \Rightarrow (t \vee s), (q \wedge t) \Rightarrow u, p, \neg s, r\} \models u$$

On met les formules sous forme de clauses :

1. $(p \vee t) \Rightarrow q \equiv \neg(p \vee t) \vee q \equiv (\neg p \wedge \neg t) \vee q \equiv (\neg p \vee q) \wedge (\neg t \vee q)$ ce qui donne deux clauses : $(\neg p \vee q)$ et $(\neg t \vee q)$
2. $r \Rightarrow (t \vee s) \equiv \neg r \vee (t \vee s) \equiv \neg r \vee t \vee s$
3. $(q \wedge t) \Rightarrow u \equiv \neg(q \wedge t) \vee u \equiv (\neg q \vee \neg t) \vee u \equiv \neg q \vee \neg t \vee u$

On applique ensuite le principe de résolution à partir des six clauses obtenues :

1. $\neg p \vee q$
2. $\neg t \vee q$
3. $\neg r \vee t \vee s$
4. $\neg q \vee \neg t \vee u$
5. p
6. $\neg s$
7. r
8. q (résolution 1 et 5 sur p)
9. $t \vee s$ (résolution 3 et 7 sur r)
10. t (résolution 6 et 9 sur s)
11. $\neg t \vee u$ (résolution 4 et 8 sur q)
12. u (résolution 10 et 11 sur t)

Il faut faire attention à ne pas prendre de raccourcis. On ne peut pas appliquer le principe sur deux variables en même temps. Si on a

$$C_1 = (p \vee \neg q \vee \neg r \vee s)$$

$$C_2 = (q \vee \neg p \vee t)$$

n on peut déduire

$$\neg q \vee \neg r \vee s \vee q \vee t \text{ (résolution sur p)}$$

ou

$$p \vee \neg r \vee s \vee \neg p \vee t \text{ (résolution sur q)}$$

Complétude et Correction

Théorie des modèles	Théorie de la démonstration
Interprétation sémantique sur un domaine	*Interprétation syntaxique*
Tables de vérité des connecteurs et prédicats	*Axiomes, règles d'inférence*
Tautologie	*Théorème*
Conséquence $\models$	*Déduction* $\vdash$

- Un système est complet ssi $\models$ g implique $\vdash$ g (on peut démontrer toutes les tautologies)
- un système est correct ssi $\vdash$ g implique $\models$ g (tous les théorèmes sont des tautologies)